MzL728-240128 LCD 模块

编程手册

V1.0 – 2008.06



北京铭正同创科技有限公司技术资料

http://www.mzdesign.com.cn

版权声明

目　　录

1 模块简介

1.1 特点

MzL728 为一块 240×128 点阵的 LCD 显示模块，模组采用 COB 技术将 TC6963C 控制 IC 以及驱动芯片集成在模块背面的 PCB 上（包括显存），模块接口简单、操作方便；可以与各种 MCU 均可进行方便简单的接口操作。

特性简述：

1. 240 × 128 点阵 FSTN
2. 内置负压产生
3. 蓝底白字
4. 并行接口为 8080 时序 MPU 接口方式
5. 白色 LED 背光，美观大方
6. T6963C 控制器，自带西文字库

1.2 主要功能与基本参数

MzL728 显示模块的基本参数如下表：

显示模式	FSTN 液晶	蓝底白点
显示格式	240×128 点阵地图形液晶显示	
输入数据	兼容 8080 系列 MPU 数据输入	
模块尺寸	144（长）×104（宽）×14（高）mm	
视屏尺寸	114（长）× 64（宽）mm	
点大小	0.40（宽）×0.40（长）mm	
像素尺寸	0.45（宽）×0.45（长）	
背光	白色 LED	

极限电器特性：

参数	符号	最小	最大	单位	备注
供电电压 1	Vdd – Vss	-0.3	7	V	
供电电压 2	Vee – Vss	Vdd –30	Vdd + 0.3	V	
供电电压 3	V0 – Vss	Vee – 0.3	Vdd+0.3	V	
输入电压	Vi	-0.3	Vdd+0.3	V	

操作温度范围	Topr	0	50	℃	常温玻璃
贮存温度	Tstr	-20	70	℃	
操作温度范围	Topr	-20	70	℃	宽温玻璃
贮存温度	Tstr	-30	80	℃	
工作湿度	RH	-	95	%	
振动测试	-	100~300Hz X/Y/Z 方向 各 1 小时			4.9m/ss 0.5g
碰撞测试	-	10ms X/Y/Z 方向 各一次			29.4m/ss 3.0g

电器特性：

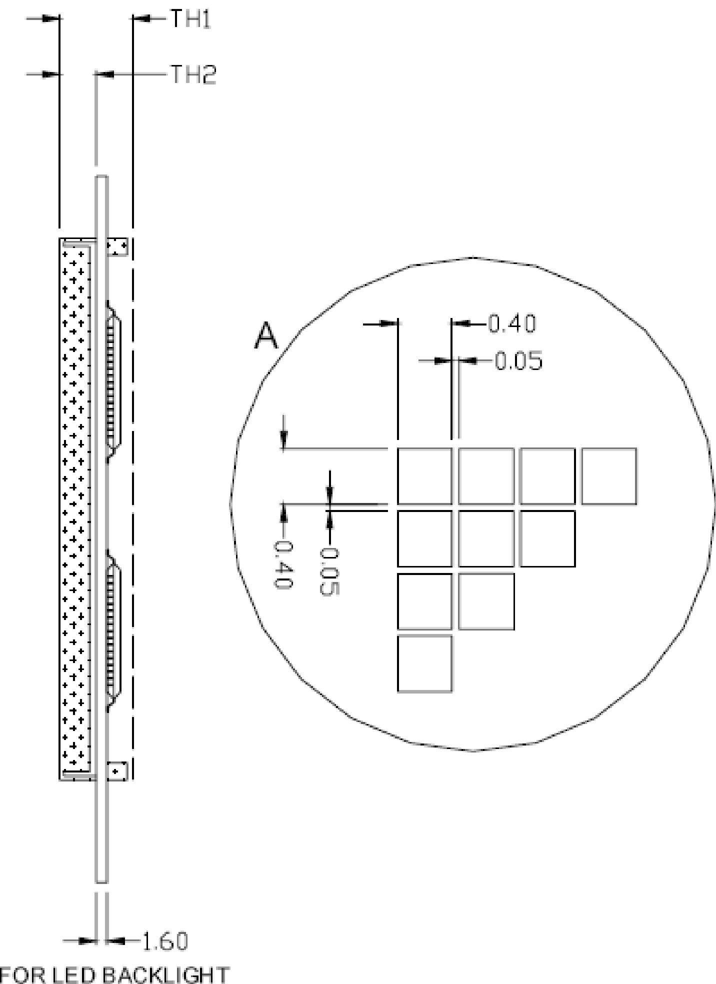
参数		符号	条件	最小	典型	最大	单位
工作电压		Vdd-Vss	—	4.5	5.0	5.5	V
供电电压		Vdd-V0	Vdd=5V	-	14.5	-	V
输入电压	High Level	Vih	高电平	Vdd-2.2	-	Vdd	V
	Low Level	Vil	低电平	0		0.8	
输出电压	High Level	Voh	高电平	Vdd-0.3	-	Vdd	V
	Low Level	Vol	低电平	0	-	0.3	V
工作电流（逻辑部分）		Idd	Vdd = 5V	-	20	-	mA
工作电流（LCD 驱动部分）		Io	Vdd-V0 = 14.5V	-	50	-	mA

背光电器特性：

参数	符号	最小	典型	最大	单位
正向电压	Vf	-	3	5	V
正向电流	If	-	200	-	mA
反向电压	Vr	-	-	8	V
功耗	Pd	-	700		mW

1.3 结构及引脚示意

图 1.1 为 MzL728 模块的结构尺寸示意图。



144.00±0.30
3.00 138.00 3.00
5.40 133.20 5.40
12.00 114.00 12.00
15.03 107.95 15.02
72.00
3.50 9.10 13.50 16.72
104.00 97.00 79.80 64.00 57.55 49.00
A
240 X 128 DOTS
2.50
20-ø1.0
1 20
3.50 15.10 19.50 22.73
PITCH 2.54
18.82 48.26
4-ø3.0

图 1.1 模块结构尺寸

1.3.1 实物照片及引脚描述



图 1.2 模块实物展示



图 1.3 实际显示效果示意

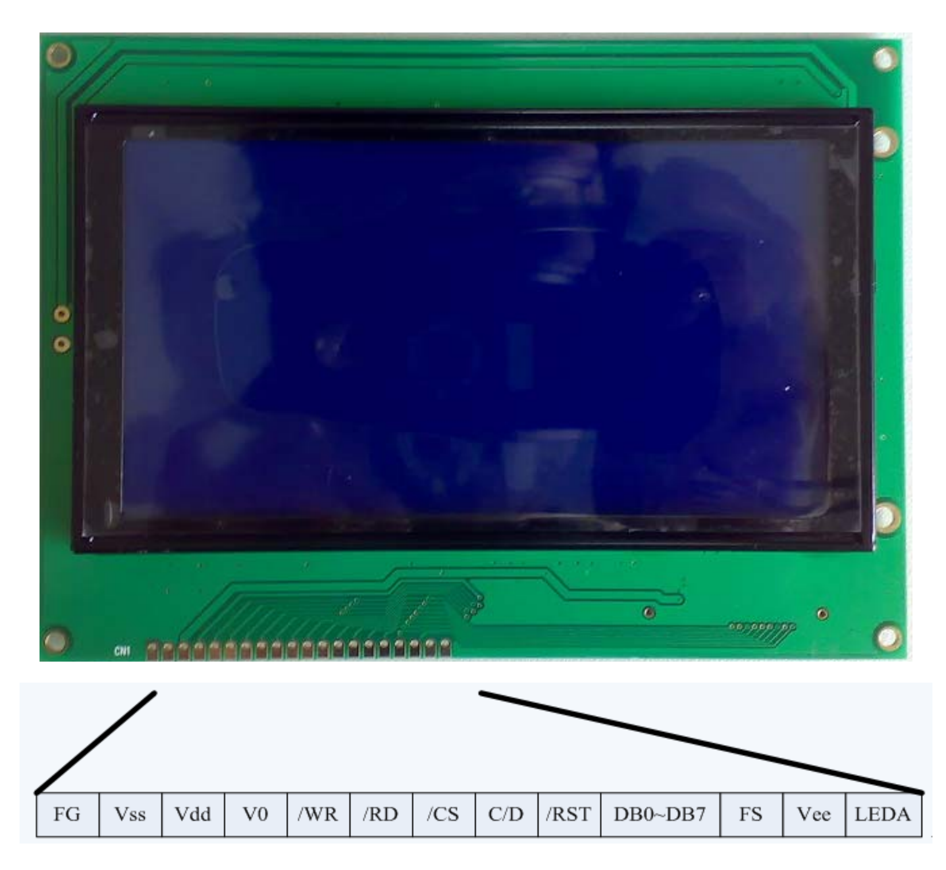


图 1.4 模块正面展示

表 1.1 模块接口引脚说明

序号	接口引脚名	说明
1	FG	外框地

2	Vss	电源地
3	Vdd	LCD 模块供电（5V）
4	V0	LCD 驱动供电调节（整屏灰度调节）
5	/WR	写控制线（低电平有效）
6	/RD	读控制线（低电平有效）
7	/CS	片选（低电平有效）
8	C/D	命令/数据选择控制，低：数据读/写　高：状态读/指令写
9	/RST	复位控制线（低电平复位）
10-17	DB0~DB7	8 位数据总线
18	FS	字体选择：低电平：8×8　高电平：6×8
19	Vee	负压输出
20	LEDA	背光正极输入

FG 外模块外部铁框的连接，一般连接到 Vss 即可，模块中背光的负极与 Vss 连接到一起了，仅需要在 LEDA 接入正极电源就可以点亮模块背光。

一般使用时用一个 10K 左右的电位器，两端接在 Vee 和 Vss 上，抽头接入 V0，调节好电位器，可以使得 LCD 模块的整屏灰度调整好。

1.4 操作时序

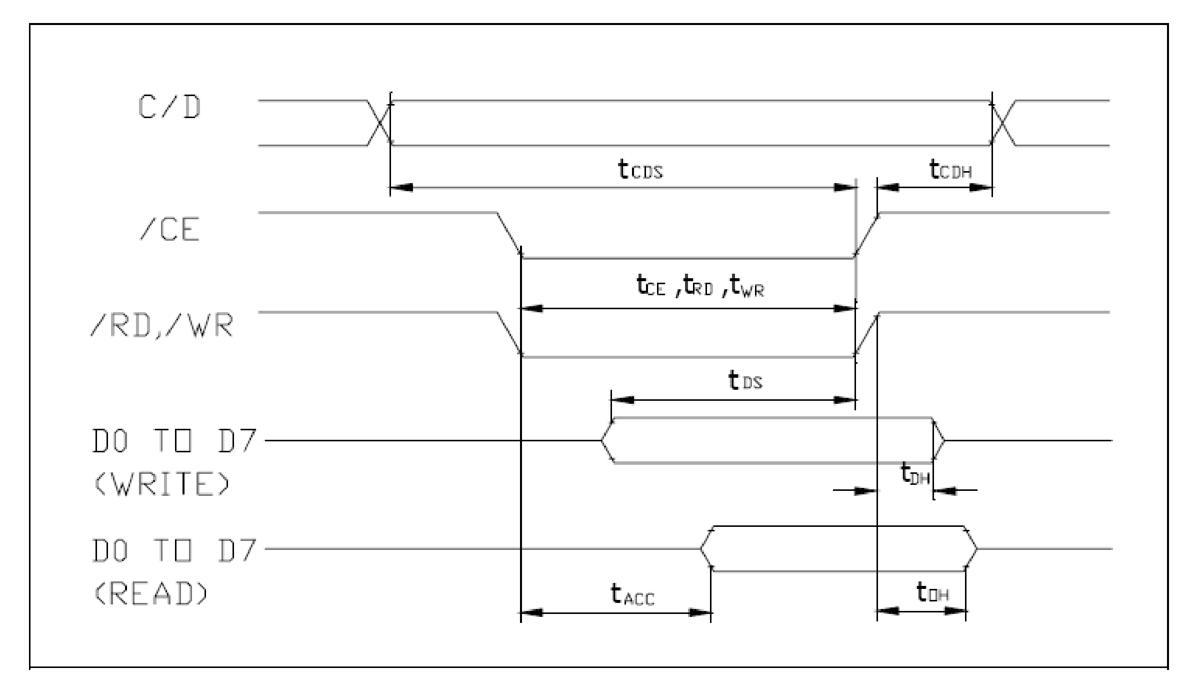


图 1.5 时序操作示意图

参数	符号	最小	最大	单位
C/D 建立时间	t_{CDS}	100	-	ns
C/D 保持时间	t_{CDH}	10	-	ns
/CE，/RD，/WR 脉冲宽度	t_{CE}、t_{RD}、t_{WR}	80	-	ns
数据建立时间	t_{DS}	80	-	ns
数据保持时间	t_{DH}	40	-	ns
数据载入时间	t_{ACC}	-	150	ns
输出保持时间	t_{OH}	10	50	ns



2 液晶显示器介绍

MzL728 模块采用 T6963C 作为 LCD 的控制器，所以液晶屏上的点阵与 LCD 控制器连接的显存的对应关系与其它的同样控制器的液晶模块类似；T6963C 可以同时支持字符层（自带的西文字库以及 CGRAM 字库显示层）和图形点阵层的显示，两者的叠加显示关系可由控制指令来设置，请参考后面对该控制器的指令的介绍；而字符层和图形点阵层的显存都放置在控制器连接的显存芯片上，其具体的位置与用户通过指令的设置相关，这里仅对液晶显示器（LCD 屏）与显存的相对对应关系作出介绍。另外，用户可以通过自定义的字库 CGRAM 也是存放在显存芯片上。

此外，显存与 LCD 屏上的点的对应关系还与 LCD 模块上的 FS 字符大小选择脚上的电平有关系。

2.1 图形点阵层显示 RAM 区映射情况

MzL728 模块的图形点阵层是可以通过指令来设置其起始地址（SDADDR）以及每行显示的宽度（每行多少个 bytes 的显存占用，WDADDR）；当 FS 脚的电平为低电平时，每个在图形点阵层的显存单元（1 个 byte）对应着 LCD 屏上的 8 个点；而当 FS 脚为高电平时，每个 byte 对应的是屏上的 6 个点，这时每个 byte 的高两位是无效的（也即不对应着 LCD 屏上的点）。

当然，该模块的屏上每行仅有 240 个点，也就是在 FS 为低时，仅能对应着 30 个 byte 的显存；所以如果 WDADDR 设置的宽度大于 30 时，多出来的显存是不会对应对 LCD 屏上的，也就是说用户是需要根据自己设置的 WDADDR 的值来计算正确的显存单元位置以对应上 LCD 屏上的具体的点。

驱动控制芯片的显示RAM区每个byte的数据对应屏上的点的排列方式为：横向排列，高位在前低位在后；如图 2.1所示：

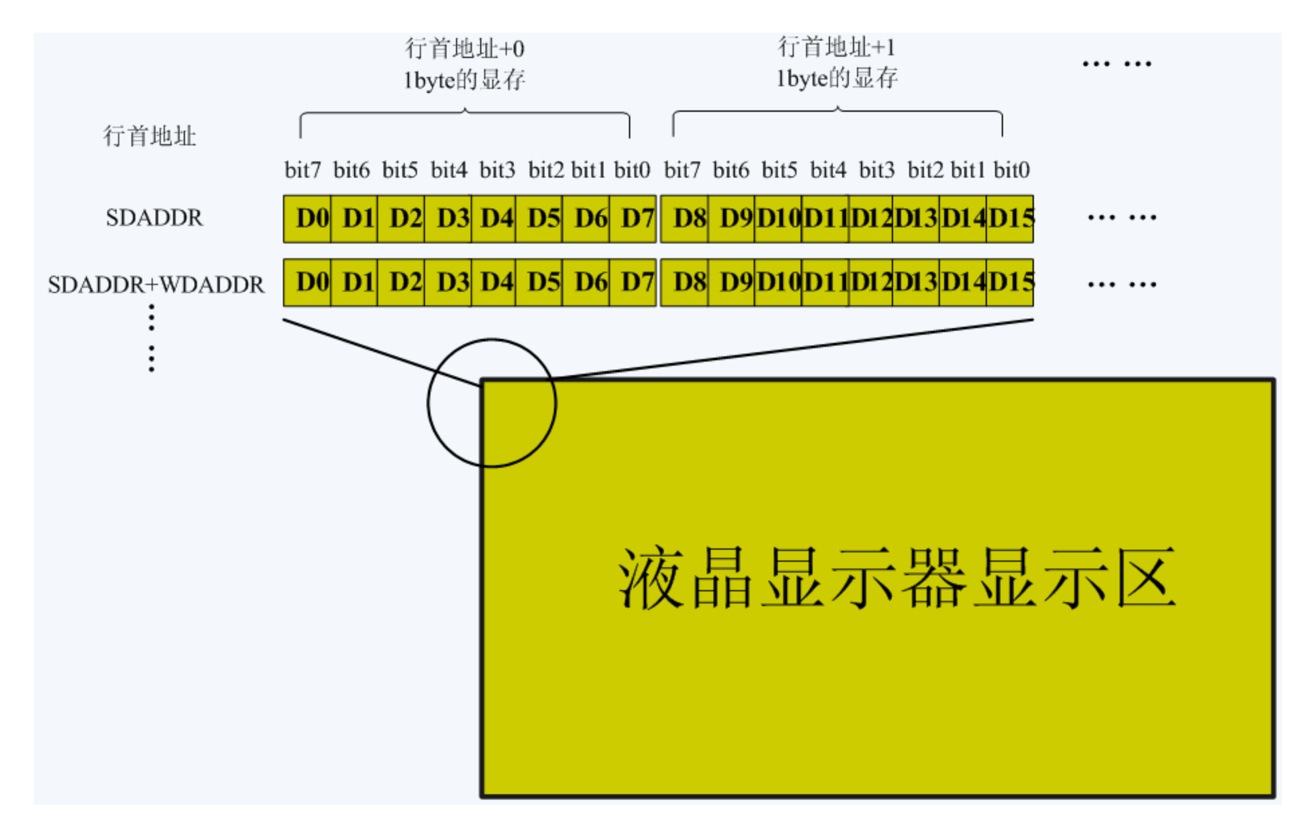


图 2.1 字节数据排列情况 1

而当 FS 脚为高电平时，每个显存的 byte 则仅对应着 6 个 LCD 屏上的点，高两位的数据是无效的；如下图所示：

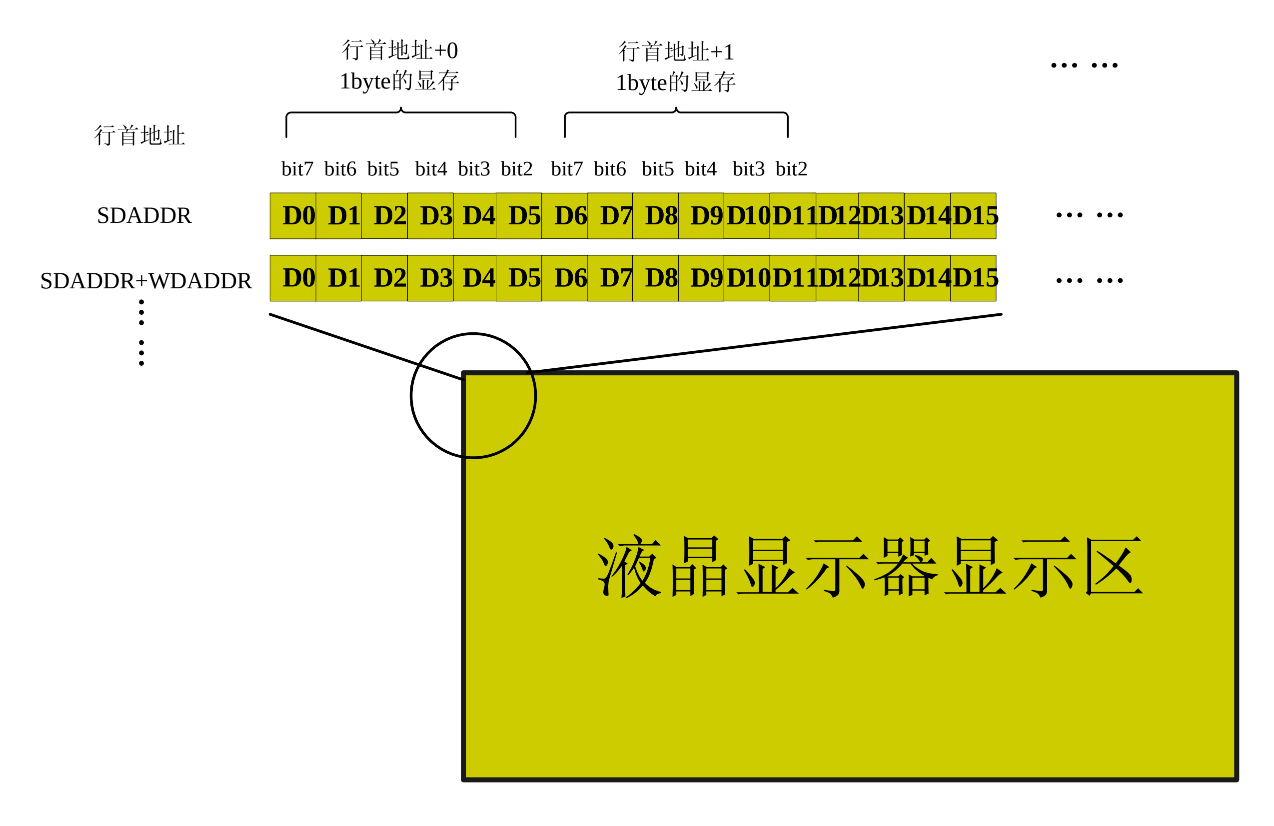


图 2.2 字节数据对应排列 2

2.2 字符层显示 RAM 区映射情况

字符层的使用类似于一些字符 LCD 模块，可以定义为是控制器自带的西文字库的字符，也可以是用户自定义在显存中的 CGRAM 字符（需要通过指令来选择，两个字符库不可同时使用）；自带的西文字符的编号从 0x00 到 0x7F，而自定义的 CGRAM 中定义的字符编号排列在 0x00~0xFF 当中；每个字符由一个显存 RAM 中的 byte 表示，控制着 LCD 屏上的 8×8 个点的显示（FS 为低电平时为 8×8 点，而 FS 为高电平时为 6×8 个点）。

字符层的显存起始地址（SCADDR）和每行宽度（WCADDR）可以通过设置指令来设置；当 FS 为低电平时，每个字符层的显存 byte 控制着对应着 8×8 点的 LCD 屏点阵显示；当然，该字节对应的 LCD 屏的显示区域实际显示字符与该字节的值来决定，当该用户选择使用控制器内部的字符库时，字节的值为 0x00 到 0x7F 之间时，为控制器自带的西文字库，有点类似于 ASCII 字符的排列；而当用户选择使用自定义在显存中的 CGRAM 字符库时，其值可为 0x00 至 0xFF 之间时，该字符由用户写入的 CGRAM 中的自定义字库的对应字模来决定。

自定义的 CGRAM 也是在控制器连接的显存芯片当中划分出来的，其起始地址（SGADDR）是可以通过指令来设置的。

字符层显存与 LCD 屏的映射关系如下图所示：



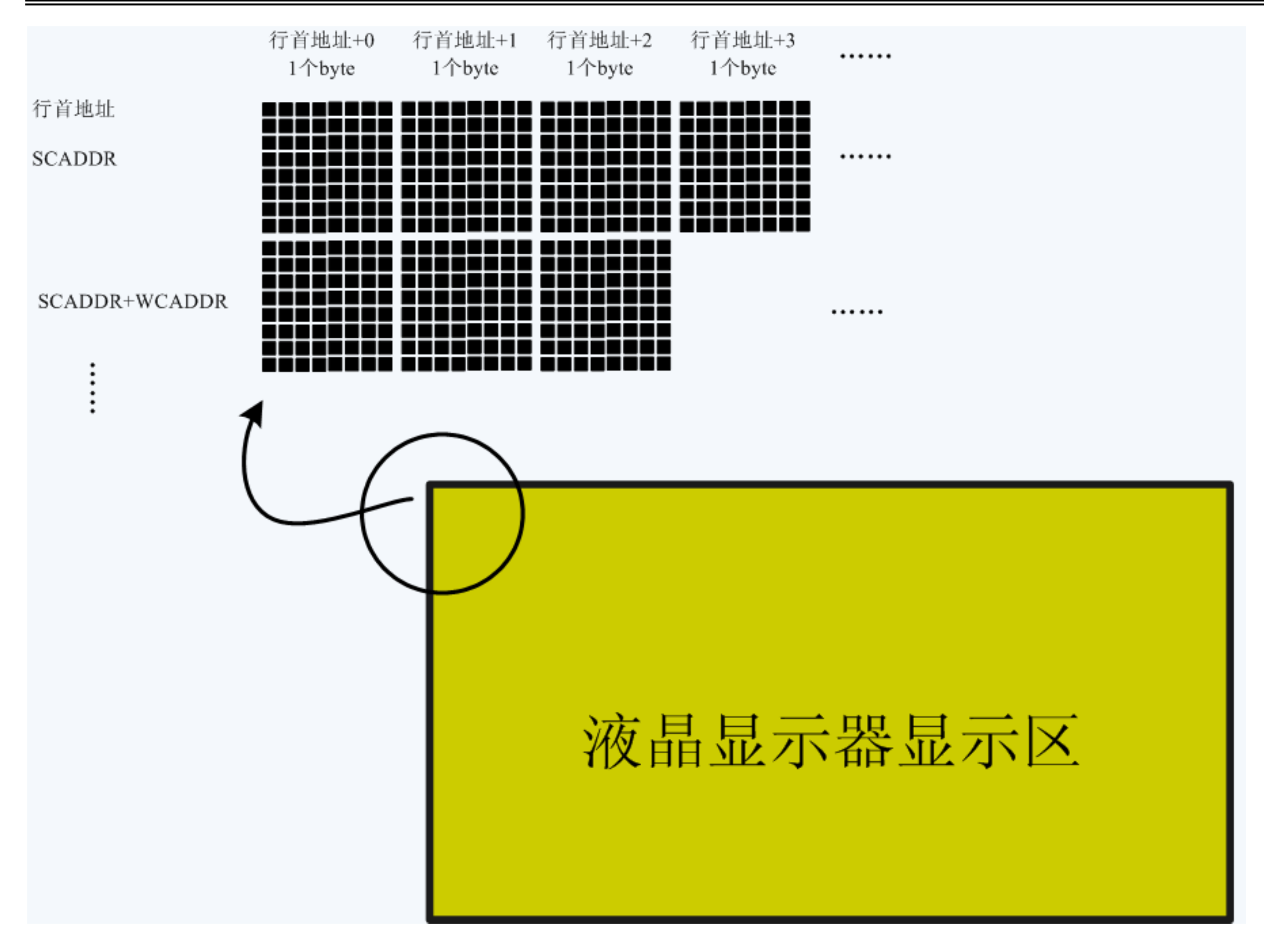


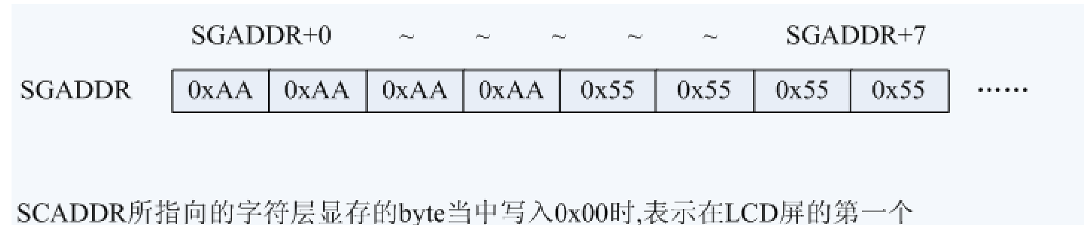
图 2.3 字符层显存映射示意 1（FS 为低电平）

当 FS 接高电平时，字符为 6×8 的点阵规模，这里就不作过多陈述。

2.3 CGRAM 用户自定义字符库字模存储区

T6963C 允许用户定义 256 个自定义的字符，存放于 CGRAM 当中，当然也是放置在连接的显存上，它的具体的位置（起始地址 SGADDR）可以由用户通过指令来设置，最大管理 2K byte 的 CGRAM。

在使用自定义在显存中的 CGRAM 时，每 8 个 bytes 的空间为一个自定义字符的字模，可以管 8×8 点阵的 LCD 屏区域，其排列方式与图形点阵层类似，比如下图，在 SGADDR 起始的连续 8 个 bytes 写入了 0xAA 和 0x55，则当在字符显示层的字符编号为 0x00 时，LCD 控制器将从 SGADDR 处提取这 8 个 bytes 的字模在 LCD 屏上显示；当然这里所述的前提是 FS 为低电平。



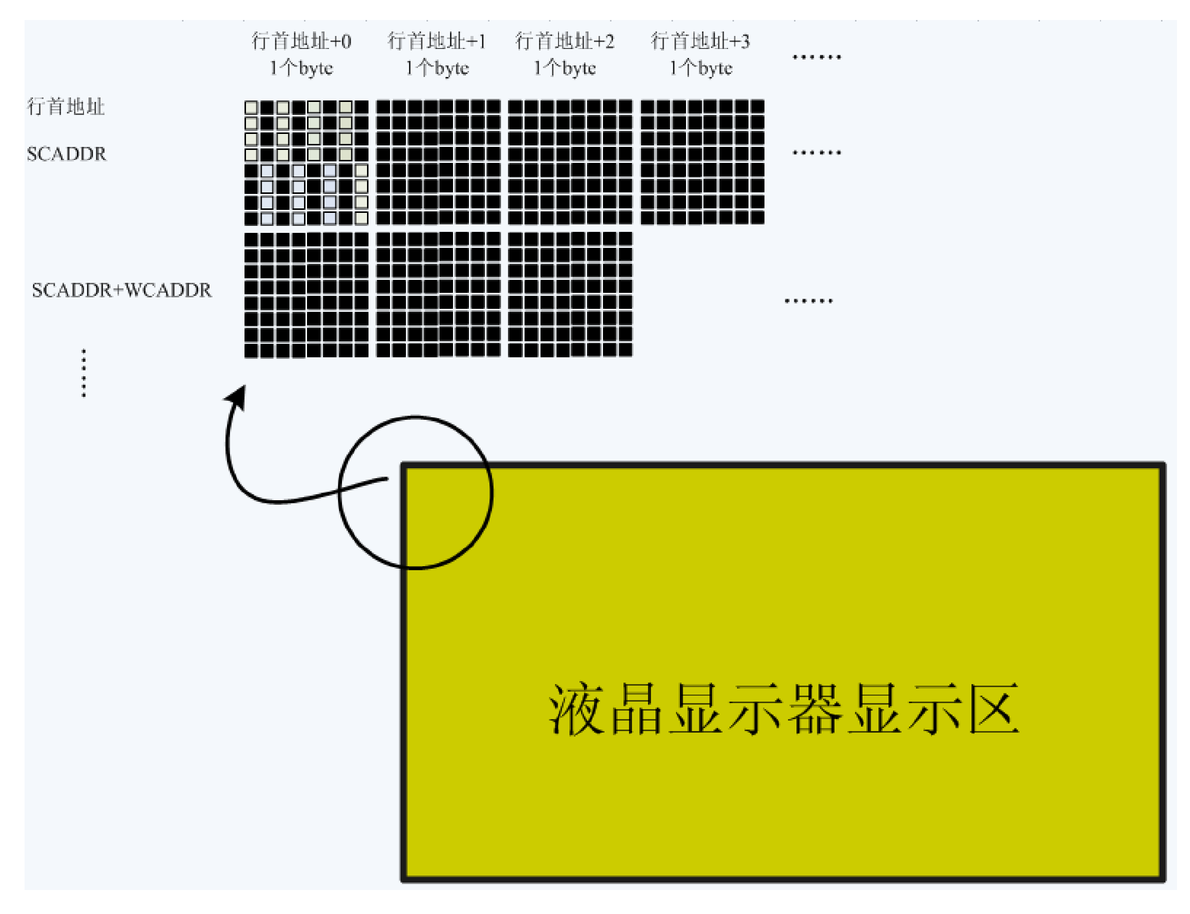


图 2.4 CGRAM

CGRAM 当中，总是 8 个 bytes 为一个自定义字符的字模数据存储区，也就是每 8 个 bytes 对应着一个从 0x00 到 0xFF 之间的字符编号。

2.4 显存

模块内有 32Kbytes 的 SRAM 显存，地址为 0x0000~0x7FFF，所有的字符层显存、图形点阵层以及 CGRAM 的数据都放置在这 32K 的 SRAM 当中。用户可以用过指令来初始化设置各个数据段的存放起始位置。

2.5 控制方法及 LCD 显示特性

T6963C 的 LCD 控制器在每一次数据/指令操作前，都要求查询控制器的状态位，以确定当前是否可以对控制器进行数据或者指令的操作；而 T6963C 的操作比较特殊，通常都是先输入数据，然后再输入指令，这些都是由具体的指令来定的，比如有的指令需要带 2 个指令数据，则在输入时就需用先输入 2 个字节的数据，最后才输入指令；当然也有一些指令是不需要带指令数据的。

2.5.1 状态字读取

当 C/D=1（高电平）、/RD=0、/WR=1、/CE=0 时，可以通过 8 位数据总线读取 LCD 控制器的状态位信息，状态位的标识及各个位的意义如下图及表格所示：

MSB LSB

STA7	STA6	STA5	STA4	STA3	STA2	STA1	STA0
D7	D6	D5	D4	D3	D2	D1	D0

STA0	指令写入状态	0：忙	1：准备好
STA1	数据读写状态	0：忙	1：准备好
STA2	数据自动读状态	0：忙	1：准备好
STA3	数据自动写状态	0：忙	1：准备好
STA4	保留		
STA5	检查控制器操作状态	0：禁止	1：使能
STA6	屏读/屏写错误标识	0：正确	1：错误
STA7	闪烁状态标识	0：显示关	1：显示正常

以上表格中的 7 个状态位并不会在同一时有内有效，它们各有各的作用场合；STA0 和 STA1 通常要同时检测，用在对控制器进行数据读/写、指令写入之前，以确定 LCD 控制器当前可以处理指令及数据，STA0 和 STA1 要同时有效，即表示控制器当前已经准备好了等待操作。而当控制器工作在连续读/写数据的状态之下时，STA2 和 STA3 状态位替代了 STA0 和 STA1 的作用。

STA6 用于指示控制器进行屏读/写的状态，STA5 和 STA7 用于指示 LCD 控制器的内部工作状态，通常在应用中并不使用它们。

T6963C 控制器在每一次写入/读出操作前，都要读取相对应的状态位，以确定控制器当前不处于忙的状态，否则操作将被忽略。

2.5.2 指令、数据写入方法

T6963C 控制器的指令当中，有的指令是需要有指令参数的（即指令数据），有的是两个字节，有的是一个字节，也有的指令是无参的；在每个指令/数据的写入操作前，都需要对相应的状态位进行判断，检查控制器当前是否处于忙的状态之下，如果控制器准备好了，就可以进行指令/数据的写入操作了。

如果在写入指令前，写入的数据多于该指令的参数个数，则控制器会取根据该指令的参数个数对应的最近的数据输入作为指令的参数。下图为操作流程图：

图 2.5 指令写入操作流程图

2.5.3 控制器指令介绍

下表为 T6963C 的指令列表：

指令	指令代码	参数 Data1	参数 Data2	功能描述
地址指针设置	0010 0001 0010 0010 0010 0100	X 轴地址 参数 地址低字节	Y 轴地址 0x00 地址高字节	设置光标位置 设置 CGRAM 偏移地址 设置显存操作地址指针
显示区域设置	0100 0000 0100 0001 0100 0010 0100 0011	地址低字节 参数数值 地址低字节 参数数值	地址高字节 0x00 地址高字节 0x00	设置字符层显存起始地址 设置字符层一行显存数 设置图形点阵层起始地址 设置图形点阵层一行显存数
显示方式设置	1000 x000 1000 x001 1000 x011 1000 x100 1000 0xxx 1000 1xxx	- - - - - -	- - - - - -	或叠加显示模式 异或叠加显示模式 与叠加显示模式 字符属性模式 控制器内置 CGRAM 模式 控制器外置 CGRAM 模式
显示状态设置	1001 $N_3N_2N_1N_0$	-	-	N_3：图形点阵层显示开关 N_2：字符层显示关关 N_1：光标显示开关 N_0：光标闪烁开关 N_x=0—关；N_x=1—开
光标形状设置	1010 $0N_2N_1N_0$	-	-	$N_2N_1N_0$=n（n=0~7） 光标设置为 8 点×(n+1)宽度
数据自动读/写	1011 0000 1011 0001 1011 0010	- - -	- - -	设置进入自动写状态 设置进入自动读状态 自动读写结束
数据读/写操作	1100 0000 1100 0001 1100 0010	数据 - 数据	- - -	写入一个字节数据，地址自加 1 读出一个字节数据，地址自加 1 写入一个字节数据，地址自减 1

	1100 0011 1100 0100 1100 0101	- 数据 -	- - -	读出一个字节数据，地址自减 1 写入一个字节数据，地址不变 读出一个字节数据，地址不变
屏读（一字节）	1110 0000	-	-	屏读出一个字节
屏拷贝（一行）	1110 1000	-	-	屏拷贝
位操作	1111 $N_3N_2N_1N_0$	-	-	N_3：0—位清零　1—置位 $N_2N_1N_0$=n（n=0~7） 对当前地址的第 n 位进行操作

1、地址指针设置指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
地址指针设置	0010 0001 0010 0010 0010 0100	X 轴地址 参数 地址低字节	Y 轴地址 0x00 地址高字节	设置光标位置 设置 CGRAM 偏移地址 设置显存操作地址指针

当指令码为 0x21 时，该指令为光标位置设置指令，通过其两个参数来指令光标的位置；在 T6963C 当中光标的地址是与当前操作的地址指针独立开的，X 轴的位置可以从 0~0x4F(80 个字符)，表示从屏幕的左起第几个字符位置，Y 轴的位置可以从 0~0x1F，表示从屏幕上方开始往下第几个字符位置；当然 MzL728 模块的点阵数为 240×128 点，实际上每行、每列并没有那么多的字符，所以在该模块当中 X 轴位置有效值为 0~0x27，Y 轴位置有效值为 0~0x15。

当指令码为 0x22 时，该指令为设置外置在显存中的 CGRAM 的首地址偏移位置的，这样经设置后，控制器在使用用户自定义的 CGRAM 字符库时，才能正确的找到存放在 CGRAM 当中的自定义字符库及其中的字模。该指令为单参数指令，参数中的低 5 位作为 CGRAM 在显存中的 16 位偏移地址的 bit15~bit11 位。比如，输入了 0x03-0x00-0x22 的指令及指令参数，则将 CGRAM 在显存中的首地址设置为了 0x1C00，如果设置了选用外置的显存 CGRAM 字符发生器，当在字符层中某个字符取编号为 0x00 时，控制器将会从 0x1C00 开始取 8 个 bytes 的字模数据在屏上进行显示。

当指令码为 0x24 时，该指令为显存地址指针设置指令；该地址为 16 位长度，分别由两个字节的指令参数进行指定。

2、显示区域设置指令

该指令为双参数指令，可以用于设置显存当中字符层、图形点阵层的起始位置，以及它们的宽度。

指令	指令代码	参数 Data1	参数 Data2	功能描述
显示区域设置	0100 0000	地址低字节	地址高字节	设置字符层显存起始地址
	0100 0001	参数数值	0x00	设置字符层一行显存数
	0100 0010	地址低字节	地址高字节	设置图形点阵层起始地址
	0100 0011	参数数值	0x00	设置图形点阵层一行显存数

当指令码为 0x40 时，为设置字符层显存的起始地址；该位置对应着屏幕左上方第一个字符的位置。

当指令码为 0x41 时，为设置字符层一行占用的显存数量；该指令的设置与字符层首地址的设置决定了字符显示模式下显存与 LCD 屏幕点的对应关系。当设置的一行显存数超过屏幕一行所能容下的字符数时，超出部分将不在屏上显示。该指令所设置的值在这里定义为 WCADDR，请参考上一章中关于字符层显存映射图。

当指令码为 0x42 时，为设置图形点阵层在显存中的起始地址（SDADDR）；该位置的字节数据对应着屏幕左上方开始的一行中的 8 个点的映射。

当指令码为 0x43 时，为设置图形点阵层一行占用的显存单元数量；该指令的设置（WDADDR）与图形点阵层起始地址的指令设置决定了图形点阵显示模式下显存与 LCD 屏中的点的映射关系。当设置的一行显存数超过屏幕一行所能容下的点数对应的字节数，则超出部分将不在屏幕上显示。具体可以参考上一章中关于图形点阵层的显存映射图。

3、显示方式设置指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
显示方式设置	1000 x000	-	-	或叠加显示模式
	1000 x001	-	-	异或叠加显示模式
	1000 x011	-	-	与叠加显示模式
	1000 x100	-	-	字符属性模式
	1000 0xxx	-	-	控制器内置 CGRAM 模式
	1000 1xxx	-	-	控制器外置 CGRAM 模式

该指令为无参数指令，其低三位的值决定图形点阵层与字符层的叠加显示关系，当然要想呈现出这两个层的叠加显示效果，需要把这两个层的显示都打开。

而当低三位值为 100 时，其特性比较特殊，当设置了字符属性模式后，图形点阵层的显存将被字符层的字符属性定义所占用，其地址对应关系将与字符层所定义的相同；即字符层显存当中仍旧用于保存为控制屏上显示字符的字符编号，而这时在图形点阵层的显存中，对应着同样偏移位置的字符的字符属性，这一共两个字节的数据将控制着屏幕上字符的显示。

字符属性模式下，每个字符的属性存放在原图形点阵层的对应地址的显存当中，由一个字节的 4 个位来定义，如下：

X	X	X	X	b3	b2	b1	b0

bit3 位定义所对应的字符的闪烁属性，0 为不闪烁，1 为字符闪烁。

bit2~bit0 定义了对应字符的显示属性，其值所对应的属性如下表所示：

b2　b1　b0	显示属性
0　0　0	正向显示
1　0　1	反向显示
0　1　1	禁止显示

该指令的指令码中的 bit3 位控制 CGRAM 的选择，即字符库的选择，当该位为 0 时选择控制器内置的字符发生器 CGRAM，允许的字符编码为 0~0x7F，一共 128 个控制器内置的字符；当该位为 1 时，选择外置在显存中的字符发生器 CGRAM，允许的字符编码为 0~0xFF，共 256 个字符，当然这些字符的字模需要事先存放到显存中的 CGRAM 当中。

4、显示状态设置指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
显示状态设置	1001 $N_3N_2N_1N_0$	-	-	N_3：图形点阵层显示开关 N_2：字符层显示关关 N_1：光标显示开关 N_0：光标闪烁开关 N_x=0—关；N_x=1—开

显示状态设置指令为无参数指令，其指令码的低 4 位分别控制着光标闪烁、光标显示、字符层显示以及图形点阵层显示的开与关。

5、光标形状设置指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
光标形状设置	1010 $0N_2N_1N_0$	-	-	$N_2N_1N_0$=n（n=0~7） 光标设置为 8 点×(n+1)宽度

该指令可以设置光标的点数，指令码中的低 3 位数值指示了光标将占用的行数。

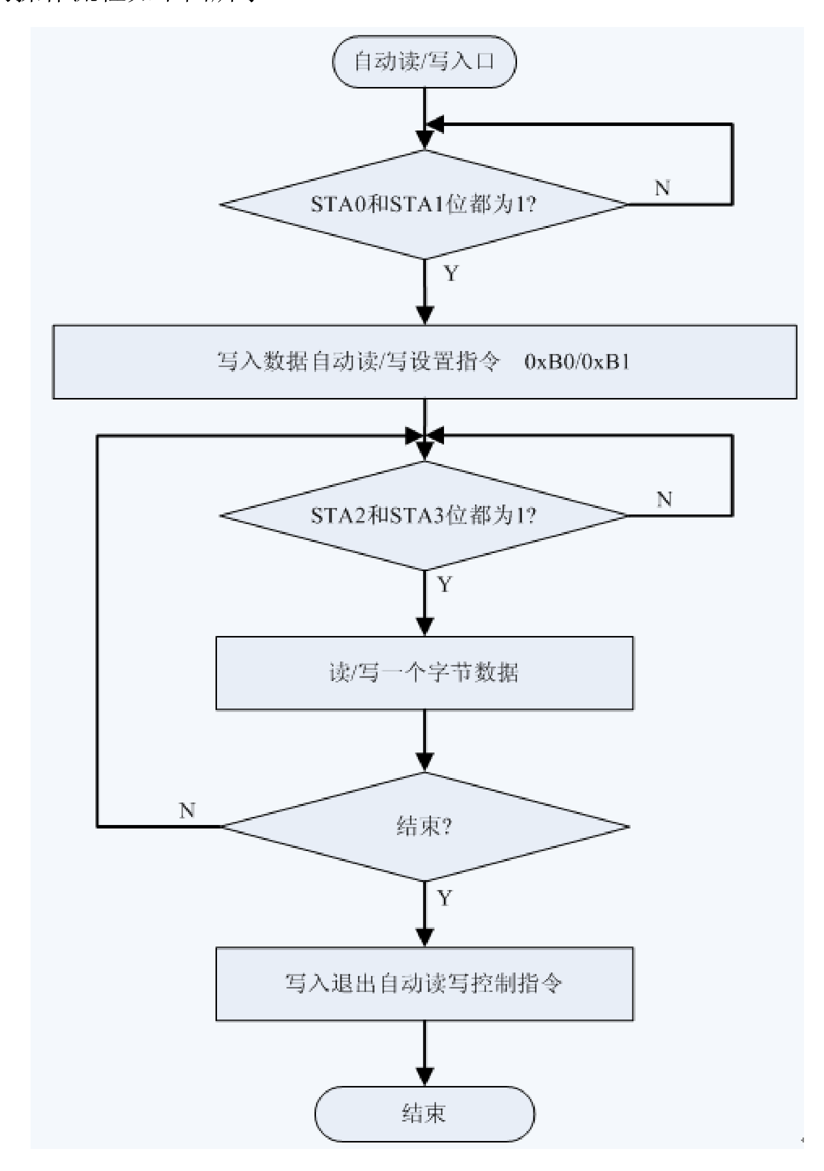
6、数据自动读/写操作指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
数据自动读/写	1011 0000	-	-	设置进入自动写状态
	1011 0001	-	-	设置进入自动读状态
	1011 0010	-	-	自动读写结束

该指令用于设置控制器的数据自动读/写控制，在自动读/写数据的模式设置后，可以连续的读取显存数据，或连续的把数据写入显存当中，地址指针会在每次操作后自动加 1；在进入自动读/写模式后，每次数据操作前，都需要判断状态位的 STA2 和 STA3 位，以确定当前控制器已经准备好了接受用户的控制。

在自动读/写操作完成后，需要写入自动读写结束指令，即写入指令 0xB2，以结速自动读/写模式。

数据自动读/写的操作流程如下图所示：



7、数据读/写操作控制指令

该指令为控制一次读/写数据操作的指令，可以选择不同的读写方式，即读写后显存地址指针的修改方式，如下：

指令	指令代码	参数 Data1	参数 Data2	功能描述
数据读/写操作	1100 0000	数据	-	写入一个字节数据，地址自加 1
	1100 0001	-	-	读出一个字节数据，地址自加 1
	1100 0010	数据	-	写入一个字节数据，地址自减 1
	1100 0011	-	-	读出一个字节数据，地址自减 1
	1100 0100	数据	-	写入一个字节数据，地址不变
	1100 0101	-	-	读出一个字节数据，地址不变

8、屏读一字节数据控制指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
屏读（一字节）	1110 0000	-	-	屏读出一个字节

屏读指令可以将屏幕上显示的图形点阵信息或者图形点阵层与字符层叠加显示的信息读回，但需要地址指针指向图形点阵层的显存区域；写入屏读指令后，需要判断状态位中的 STA6，以确定该指令能够正确执行了，如 STA6 为 1，则可以进行数据的读取操作。

9、屏拷贝一行数据控制指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
屏拷贝	1110 1000	-	-	屏拷贝

屏拷贝指令可以将屏幕上显示的图形点阵信息或者图形点阵层与字符层叠加显示的信息读回，可以一次设置后读回一行的 LCD 点数所对应的字节数据，但需要地址指针指向图形点阵层的显存区域；写入屏读指令后，需要判断状态位中的 STA6，以确定该指令能够正确执行了，如 STA6 为 1，则可以进行数据的读取操作。

10、位操作指令

指令	指令代码	参数 Data1	参数 Data2	功能描述
位操作	1111 $N_3N_2N_1N_0$	-	-	N_3：0—位清零　1—置位 $N_2N_1N_0$=n（n=0~7） 对当前地址的第 n 位进行操作

该指令可以对显存中指针指向的单元字节的任一位进行写 1 或清 0 操作，由指令码的 bit3 位决定位

操作是置位还是位清零，而低 3 位的数值决定要操作的位。该指令无参数。

3 设计参考

铭正同创为广大用户提供了通用版的 LCD 接口驱动程序，下面将介绍与 LCD 以及操作 LCD 的 MCU 型号类型相关的底层函数。

3.1 基于 MCS51 的 LCD 驱动：

LCD 基本接口函数：包括写数据、写指令、读数据、读状态字的子程序，如下。

```
#include "REG52.h"
#include "intrins.h"            //包含此头文件可直接操作内核的寄存器以及一些定义好的宏

sbit LCD_FS = P2^5;

sbit LCD_WR = P2^0;
sbit LCD_RD = P2^1;
sbit LCD_A0 = P2^3;
sbit LCD_RE = P2^4;
sbit LCD_CS = P2^2;

#define LCD_CMD_Dir             P2
#define LCD_CMD_Buffer          P2

#define LCD_Data_BUS_Out        P0
#define LCD_Data_BUS_In         P0
#define LCD_Data_BUS_Dir        P0
//========================================================================
// 函数: void LCD_DataWrite(unsigned int Data)
// 描述: 写一个字节的显示数据至 LCD 中的显示缓冲 RAM 当中
// 参数: Data 写入的数据
// 返回: 无
// 备注: 无
// 版本:
//      2007/01/09      First version
//========================================================================
void LCD_DataWrite(unsigned char Dat)
{
        LCD_A0 = 0;
```

```
        LCD_CS = 0;

        LCD_Data_BUS_Out = Dat;
        LCD_WR = 0;
        LCD_WR = 1;
        LCD_CS = 1;}
//========================================================================
// 函数: unsigned char LCD_DataRead(void)
// 描述: 从 LCD 中的显示缓冲 RAM 当中读一个字节的显示数据
// 参数: 无
// 返回: 读出的数据，
// 备注:
// 版本:
//      2007/01/09      First version
// 注意:
//========================================================================
unsigned char LCD_DataRead(void)
{
        LCDBYTE Read_Data;
        LCD_Data_BUS_Out = 0xff;            //
        LCD_A0 = 0;
        LCD_CS = 0;
        LCD_RD = 0;
        Read_Data = LCD_Data_BUS_In;

        LCD_RD = 1;
        LCD_CS = 1;
        return Read_Data;}
//========================================================================
// 函数: void LCD_RegWrite(unsigned char Command)
// 描述: 写一个字节的数据至 LCD 中的控制寄存器当中
// 参数: Command       写入的数据，低八位有效（byte）
// 返回: 无
// 备注:
// 版本:
//      2007/01/09      First version
//========================================================================
void LCD_RegWrite(unsigned char Command)
{
```

```
        LCD_A0 = 1;
        LCD_CS = 0;
        LCD_Data_BUS_Out = Command;
        LCD_WR = 0;
        LCD_WR = 1;
        LCD_CS = 1;
}
//========================================================================
// 函数: unsigned char LCD_ReadStatus(void)
// 描述: 读取 LCD 模块的指定状态位
// 参数: 无
// 返回: 读取到的状态位
// 备注: Mz 通用版 LCD 驱动程序 标准子函数
// 版本:
//      2007/01/09      First version
//========================================================================
unsigned char LCD_ReadStatus(void)
{
        unsigned char Status=0;
        LCD_Data_BUS_Out = 0xff;            //
        LCD_A0 = 1;
        LCD_CS = 0;
        LCD_RD = 0;
        Status = LCD_Data_BUS_In;
        LCD_RD = 1;
        LCD_CS = 1;
        return Status;
}
//========================================================================
// 函数: unsigned char LCD_TestStatus(unsigned char bitMatch)
// 描述: 测试 LCD 模块的指定状态位
// 参数: bitMatch
// 返回: 如测试有效则返回 1   如超时则返回 0
// 备注: Mz 通用版 LCD 驱动程序 标准子函数
// 版本:
//      2007/01/09      First version
//========================================================================
unsigned char LCD_TestStatus(unsigned char bitMatch)
```

```
{
        unsigned shrot Error_Counter=0;
        while((LCD_ReadStatus()&bitMatch)!=bitMatch)
        {
            Error_Counter++;
            if(Error_Counter>=1000) return 0;        //return 0   if test status bit fail
        }
        return 1;                            //return 1   as test status bit ok
}
```

而建议参考的 LCD 初始化代码如下：

```
//==========================================================================
// 函数: void LCD_PortInit(void)
// 描述: 与 LCD 连接的端口初始化代码
// 参数: 无
// 返回: 无
// 备注: Mz  通用版 LCD 驱动程序  标准子函数
// 版本:
//         2007/01/09         First version
// 注意:
//==========================================================================
void LCD_PortInit(void)
{
        LCD_Data_BUS_Dir = 0xff;
        LCD_Data_BUS_Out = 0xff;

        LCD_CS = 1;
        LCD_A0 = 1;
        LCD_RE = 1;
        LCD_WR = 1;
        LCD_RD = 1;
        LCD_FS = 0;
}
//==========================================================================
// 函数: void LCD_Init(void)
// 描述: LCD 初始化程序，在里面会完成 LCD 初始所需要设置的许多寄存器，具体如果
//           用户想了解，建议查看 DataSheet 当中各个寄存器的意义
```

```
// 参数: 无
// 返回: 无
// 备注:
// 版本:
//        2006/10/15        First version
//         2007/01/09           V1.2
//==========================================================================
//延时程序
void TimeDelay(int Time)
{
        int i;
        if(Time > 0)
        {
            for(i = 0;i < 800;i++)
            {
            }
            Time --;
        }
}
void LCD_Init(void)
{
        //LCD 驱动所使用到的端口的初始化（如果有必要的话）
        LCD_PortInit();
        LCD_RE = 0;
        TimeDelay(40);
        LCD_RE = 1;
        TimeDelay(20);
        //设置 TEXT 文本层的显存起始位置
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x00);
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x00);
        if(LCD_TestStatus(0x03))
            LCD_RegWrite(0x40);
        //设置 CGRAM 的起始位置
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x20);
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x00);
        if(LCD_TestStatus(0x03))
```

```
            LCD_RegWrite(0x41);
        //设置图形层的起始位置(本驱动仅使用图形层)
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x00);
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x08);
        if(LCD_TestStatus(0x03))
            LCD_RegWrite(0x42);
        //设置一行占用的显存数量
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x20);
        if(LCD_TestStatus(0x03))
            LCD_DataWrite(0x00);
        if(LCD_TestStatus(0x03))
            LCD_RegWrite(0x43);
        //
        if(LCD_TestStatus(0x03))
            LCD_RegWrite(0xa7);
        //
        if(LCD_TestStatus(0x03))
            LCD_RegWrite(0x80);
        //图形显示层开,其它关
        if(LCD_TestStatus(0x03))
            LCD_RegWrite(0x98);
}
```

3.2 Mz 通用版 LCD 驱动程序简介

用户可以到网站www.Mzdesign.com.cn 下载最新的通用版LCD驱动程序，一般都会为每一个LCD模块提供有参考的基于该通用版LCD驱动程序的不同MCU应用，有AVR单片机、MCS51、LPC21/22XX系列ARM7 的参考代码；为了用户更好的理解程序，在网站上提供了基于该驱动程序的一份资料《点阵LCD驱动显控原理》，这是一份还算不错的参考资料。

此外，网站上还会不定期的提供一些供用户参考使用的显示控制的例子，比如一些列表式菜单的例子、图形菜单的例子，以及一些常用的函数，这些都是基于通用版 LCD 驱动程序的，基本上与具体的 LCD 模块无关，方便于用户使用在各种应用场合。

4 技术支持

铭正同创在 LCD 显示驱动方面有着丰富的经验，并通过积累总结出通用的成熟的 LCD 驱动程序架构，可方便的移植到不同的 LCD 应用上，或者移植到不同的 MCU 应用平台上。随着产品的推出，将会推出更多的完整的驱动程序，这些代码或应用开发软件用户可以放心的使用到产品当中。

而在人机界面编程方面，铭正同创也会给用户提供更多的参考，或者为用户订制专用的设计，使铭正同创的用户在得到质优价廉的产品的同时，也获得专业的技术支持。

铭正同创努力的目标是作为一个行业的**产品+技术的专业提供者**。

4.1 联系方式

- 北京铭正同创科技有限公司
- 联系电话：**010-82970200**
- 技术 **QQ**：**644272644**
- 销售 **QQ**：**734307563**
- 公司邮箱：**mzdesign@mzdesign.com.cn**
- 公司网址：**http://www.mzdesign.com.cn**